VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV MIKROELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF MICROELECTRONICS

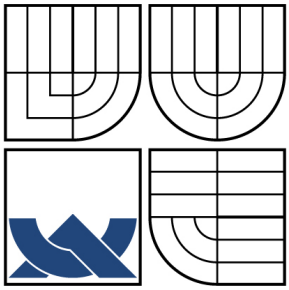
PROGRAM PRO OVĚŘENÍ A KONTROLU DOKUMENTŮ S PŘÍPONOU PDF

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

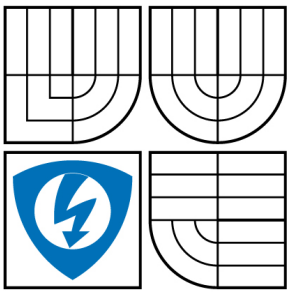
MARTIN SOHR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV MIKROELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF MICROELECTRONICS

PROGRAM PRO OVĚŘENÍ A KONTROLU DOKUMENTŮ S PŘÍPONOU PDF

PROGRAM FOR CHECKING AND VERIFYING OF PDF DOCUMENT'S CONTENT

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

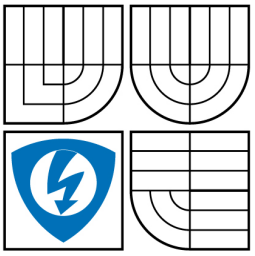
AUTOR PRÁCE — MARTIN SOHR

AUTHOR

VEDOUCÍ PRÁCE — Ing. JAROSLAV KADLEC, Ph.D.

SUPERVISOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav mikroelektroniky

Bakalářská práce

bakalářský studijní obor
Mikroelektronika a technologie

Student:	Martin Sohr	***ID:***	98187
Ročník:	3	***Akademický rok:***	2008/2009

NÁZEV TÉMATU:

Program pro ověření a kontrolu dokumentů s příponou pdf

POKYNY PRO VYPRACOVÁNÍ:

Vytvořte program pro ověření a kontrolu dokumentů pdf. Vytvořený program bude schopen zpracovávat dokumenty pdf, procházet jejich obsah a kontrolovat splnění zadaných podmínek na jeho obsah. Na základě tohoto testování bude program schopen vytvořit zprávu o obsahu jednotlivých testovaných dokumentů, který bude uložen ve formátu xml. Program bude vytvořen v programovacím jazyku C#. Výsledná bakalářská práce bude obsahovat přiložené CD s vytvořeným programem a zdrojovými kódy.

DOPORUČENÁ LITERATURA:

Podle pokynů vedoucího práce

Termín zadání:	9.2.2009	***Termín odevzdání:***	3.6.2009

Vedoucí práce: Ing. Jaroslav Kadlec, Ph.D.

prof. Ing. Radimír Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Licenční smlouva
poskytovaná k výkonu práva užít školní dílo

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Martin Sohr

Bytem: Třebíč, I. Olbrachta 652/6, 674 01

Narozen/a (datum a místo): 25.8.1986, Ostrava – Moravská Ostrava

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií

se sídlem Údolní 244/53, 602 00 Brno

jejímž jménem jedná na základě písemného pověření děkanem fakulty:

prof. Ing. Radimír Vrba, CSc.

(dále jen „nabyvatel“)

Čl. 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - □ disertační práce
   - □ diplomová práce
   - ☒ bakalářská práce
   - □ jiná práce, jejíž druh je specifikován jako .....

   (dále jen VŠKP nebo dílo)

Název VŠKP: Program pro ověření a kontrolu dokumentů s příponou pdf

Vedoucí/ školitel VŠKP: Ing. Jaroslav Kadlec, Ph.D.

Ústav: Ústav mikroelektroniky

Datum obhajoby VŠKP: 12.6.2009

VŠKP odevzdal autor nabyvateli v:

- ☒ tištěné formě – počet exemplářů 2
- ☒ elektronické formě – počet exemplářů 2

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - ☒ ihned po uzavření této smlouvy
   - □ 1 rok po uzavření této smlouvy
   - □ 3 roky po uzavření této smlouvy
   - □ 5 let po uzavření této smlouvy
   - □ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: 3. 6. 2009

…….. ……

Nabyvatel Autor

Abstrakt:

Program prochází text souboru s příponou pdf a zjišťuje informace o textu a vlastnostech kontrolovaného souboru podle zadaných parametrů. Program dokáže z metadat souboru načíst uvedené informace o autoru, titulek, obsah, klíčová slova a název programu, ve kterém byl soubor vytvořen. Z obsahu souboru je program schopen zjistit počet znaků, slov, stran a je schopen vyhledávat požadovaný text. Pro vyhledávání je možno použít hvězdičkové i otazníkové konvence. Program dokáže při jednom spuštění zkontrolovat i několik souborů.

Klíčová slova:

Program, ověření a kontrola obsahu, obsah, PDF dokument, podmínky, report, XLM soubor

Abstract:

Program runs through the text of pdf file with postfix .pdf and locates informations about text and about charakteristics of checked file based on set arguments. Program is able to load mentioned informations about author, title, content, key words and the name of program where was the file created from matadat file. Program is able to find out numbers of symbols, words, pages out of content of file and is also capable of searching for required text from the content of file. It’s possible to use star or interrogation convention for searching. Program is able to check several files in one start.

Keywords:

Program, check and inspection of content, content, pdf document, conditions, report, XLM file

Bibliografická citace mé práce:

SOHR, M. Program pro ověření a kontrolu dokumentů s příponou pdf. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 23 s. Vedoucí bakalářské práce Ing. Jaroslav Kadlec, Ph.D.

Prohlášení autora o původnosti díla:

Prohlašuji, že jsem tuto vysokoškolskou kvalifikační práci vypracoval samostatně pod vedením vedoucího bakalářské práce, s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 3. 6. 2009

…….

Poděkování:

Děkuji vedoucímu bakalářské práce Ing. Jaroslavu Kadlecovi, Ph.D. za metodické, pedagogické a cíleně orientované vedení při plnění úkolů realizovaných v návaznosti na mé bakalářské práci.

V Brně dne 3. 6. 2009

…….

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Zadání bakalářské práce spočívalo ve vytvoření *Programu pro ověření a kontrolu souboru s příponou PDF*, procházení obsahu a kontroly splněných podmínek ve vybraných souborech. Na základě tohoto je program schopen vytvořit kontrolní report o obsahu jednotlivých souborů podle zadaných podmínek kontroly.

1. PDF soubory (Portable Document Format)

První verze PDF formátu byla vyvinuta v roce 1993 firmou Adobe pod názvem "*PDF Interchange PostScript*" jako kompatibilní datový formát mezi různými OS. Výhodami byla malá velikost PDF souboru, snadné vytvoření souboru, bezplatné poskytnutí prohlížeče a vytváření dokumentů nezávisle na softwaru i hardwaru, na kterém byly pořízeny. Soubor s příponou PDF může obsahovat text i obrázky, přičemž tento formát zajišťuje, že se libovolný dokument na všech zařízeních zobrazí stejně.

1.1 Prohlížení

Pro tento formát existují freeware prohlížeče, které jsou kompatibilní s většinou operačních systémů. Mezi nejrozšířenější patří *Acrobat Reader* od firmy *Adobe*.

1.2 Tvorba

Pro vytváření souboru potřebuje uživatel program, který mu tuto funkci zprostředkuje je to takzvaný „*PDF creator*“. V současnosti je na trhu spousta takových programů. Tyto programy se většinou implementují přímo do OS, kde simulují tiskárnu (*virtuální tisk*), tím program získá přístup do většiny dnešních programů. Tímto způsobem dostává uživatel možnost přes volbu TISK vytvořit z libovolné aplikace soubor s příponou pdf. Tyto aplikace nemusí být vůbec textové (*Word, Excel, Wordpad* atd.), ale mohou to být i grafické programy (*AutoCad, 3D Studio Max, Corel* atd.), nebo libovolný program s výstupem na tiskárnu (program opatřený možností *Tisk*). Pro vytvoření dokumentu je potřeba nastavit pouze parametry tisku a výsledného požadovaného dokumentu.

Ukázka některých *PDF creatorů*:

- OpenOffice.org 2.0.3 – freeware,
- PDF-XChange – shareware,
- PDFCreator 0.9.3 – freeware,
- PDFProducer 1.3 – freeware,
- PdfFactory 3 CZE – shareware,
- Go2PDF 3.0 – freeware,
- PDF4Free 2.0 – freeware,
- PrimoPDF 2.0 – freeware,
- PDF2HTML 0.38 – freeware.

1.3 Struktura

Ve verzi 1.0 byla všechna data uložena v *7-bit ASCII* kódu. To znamená, že binární data byla zakódována v *ASCII* do sedmi bitů a řádky dokumentu musely být kratší než 255 znaků. Důvodem pro toto omezení byla snaha o korektní přenos těchto souborů.

„Postupně se ukázalo, že ani přísná omezení definovaná ve verzi 1.0 nezabránila změnám v souboru při jejich přenosu. Nežádoucí bylo také zvětšení dokumentu asi o 20%. To způsobovalo kódování binárních dat do *7-bit ASCII*. Proto se ve verzi 1.1 odstupuje od nutnosti použití tohoto formátu. V následných verzích jsou binární data uložena přímo v objektech *string*, *stream* a *v komentářích*.

Následná praxe prokázala, že soubory jsou méně poškozovány při přenosu, jestliže obsahují binární data. K tomu se používá malý trik: druhý řádek *PDF souboru* obsahuje komentář alespoň se čtyřmi znaky s *ASCII* hodnotou vyšší než 128. Tím je navozen dojem, že se jedná o binární soubor, a je s ním také tak zacházeno.

Struktura souboru PDF zůstala oproti verzi 1.0 nezměněná. Vzhledem k možnosti použití binárních dat se upustilo od požadavku na omezení délky jedné řádky. Řádky ve verzi 1.1 tedy mohou být libovolně dlouhé.“[1] Z důvodu kompatibility s nižší verzí je doporučeno nepoužívat řádky delší než 255 znaků.

1.4 Vývoj formátu

Vývoj *formátu PDF* a prohlížeče *Acrobat Reader* prezentuje tabulka (Tab. 1).

Tab. 1: Verze souborů s příponou PDF

Verze	Rok vydání	Originální prohlížeč	Upgrade proti předchozí verzi
PDF 1.0	1993	Acrobat Reader 1.0	Komptabilita platforem Windows, UNIX, Mac.
PDF 1.1	1994	Acrobat Reader 2.0	Umožnění uložení obrázků ve formátu JPEG
PDF 1.2	1996	Acrobat Reader 3.0	Podpora komprese ZIP Podpora barevného modelu CMYK
PDF 1.3	1999	Acrobat Reader 4.0	Podpora různých formátů dokumentů Nové grafické funkce Funkce ZOOM
PDF 1.4	2001	Acrobat Reader 5.0	Nová grafická vlastnost - průhlednost
PDF 1.5	2003	Acrobat Reader 6.0	Podpora komprese JPEG2000
PDF 1.6	2005	Acrobat Reader 7.0	Podpora fontů OpenType
PDF 1.7	2006	Acrobat Reader 8.0	Vkládání 3D dat

1.5 PDF v dnešní době

Vlastník patentu k *souborům s příponou pdf,* společnost *Adobe,* se rozhodla zažádat o standardizaci PDF a vytvořit tak *PDF ISO standard*. *ISO certifikát* byl společnosti *Adobe* udělen po schválení formátu PDF jako mezinárodního standardu pro publikaci elektronických dokumentů. 1. července 2008 byl formát PDF 1.7 uveřejněn jako *Standard ISO 32000-1:2008*. Tímto *společnost Adobe* významně napomohla tomu, aby se specifikace PDF dostala zdarma do rukou vývojářů, kteří pak tvoří odpovídající aplikace pro práci s tímto formátem.

1.6 Členění souboru

PDF dokument se skládá ze čtyř základních částí:

- *header* - jednořádková hlavička,
- *body* - tělo dokumentu,
- *cross reference table* - tabulka křížových odkazů,
- *trailer* – zápatí dokumentu.

Rozložení základních částí formátu PDF (Obr. 1)

<PDFFile>::=	// Název souboru PDF
<header>	// Hlavička
<body>	// Tělo dokumentu
<cross-reference table>	// Tabulka křížových odkazů
<trailer>	// Pata dokumentu

Obr. 1: Rozložení formátu PDF

1.6.1 Hlavička

Hlavička PDF – *header* - souboru zabírá prvních 1024 bitu souboru. Tento datový rezervovaný prostor je rozdělen na 2 části:

- První část definuje verzi PDF souboru s možným formátováním:
  - *%PDF-X.y*
  - *%!PS-Adobe-X.y*

    kde *X* číslo je verze, *Y* číslo je podverze (*X.y* je např. 1.1)
- Druhá část jsou binární znaky z *8-bit ASCII* tabulky, pro zajištění binárního přenosu.

1.6.2 Tělo

Tělo – *body* - (struktura) PDF dokumentu je posloupnost objektů (*fonty, stránky, obrázky*), které popisují celkový vzhled PDF dokumentů. Objevují se zde i *komentáře*, které jsou na začátku řádku uvozeny znakem % a ukončeny koncem řádku. Mohou být kdekoli v dokumentu.

1.6.3 Tabulka křížových odkazů

„Tabulka křížových odkazů - *cross-reference table* - obsahuje informace, které umožňují rychlý přístup k nepřímým objektům v souboru. Pro každý nepřímý objekt souboru, obsahuje tabulka jeden řádek, v němž je popsáno umístění objektu. Každý PDF dokument obsahuje jednu *tabulku křížových odkazů*. Tabulka se může skládat z více sekcí. Jestliže v souboru nebyly prováděny žádné modifikace, obsahuje dokument jen jednu sekci. S každou provedenou úpravou dokumentu, se přidá nová sekce *do tabulky křížových odkazů*. Sekce tabulky má přesně určený formát (viz Obr. 2). Její definice začíná klíčovým slovem *xref*. Za ním je jedna nebo více podsekcí.“[1]

<cross-reference section>:==	// Začátek sekce křížových odkazů
Xref	// Definice sekce uvozena klíčovým slovem
<cross-reference subsection>+	// Definice nové podsekce křížových odkazů

Obr. 2: Formát sekce tabulky křížových odkazů

„Do nové sekce jsou vždy přidány odkazy pouze na objekty, které byly modifikovány nebo odstraněny. Na začátku každé podsekce je *hlavička* obsahující dvě položky (Obr. 3):

- číslo prvního objektu v podsekci,
- počet objektů, na které se v podsekci odkazuje.

Za *hlavičkou* jsou umístěny jednotlivé odkazy, přičemž každý je vždy na jednom řádku.“[1]

<cross-reference subsection>:==	// Začátek podsekce křížových odkazů
<object number of first entry in subsection>	// Číslo prvního objektu v podsekci
<number of entries in subsection>	// Počet objektů v podsekci
<cross-reference entry>+	// Jednotlivé odkazy na objekty

Obr. 3: Formát hlavičky uvozující podsekci

Každý záznam v podsekci má přesně 20 znaků (i s ukončení řádku). Existují dva formáty záznamů:

1. označuje objekt, který je v dokumentu používán - *in-use entry*,
2. druhý odkazuje na odstraněný objekt - *free entry*:.

„První záznam v tabulce *křížových odkazů* (objekt číslo 0) je vždy prázdný a má generační číslo 65535. Tento záznam je první položkou v seznamu smazaných objektů. Poslední záznam v seznamu používá číslo 0, jako odkaz na další smazaný objekt.

Jestliže je při editaci dokumentu vymazán nepřímý objekt, je v *tabulce křížových odkazů* označen odkaz na tento objekt jako na smazaný. Zároveň je generační číslo v záznamu zvětšeno o jedna. Pokud je objekt se stejným číslem znovu vytvořen, je mu přiřazeno toto zvětšené generační číslo. Maximální hodnota generačního čísla je 65535. Jakmile objekt dosáhne této maximální hodnoty, není jej možné v dokumentu nadále používat.“[1]

1.6.4 Zápatí

Zápatí – *trailer* - je poslední částí souboru PDF, ale zpracovává se jako druhé (hned po *hlavičce*). Slouží k rychlému nalezení *tabulky křížových odkazů* a objektů v dokumentu. Obsahuje informace o poslední *aktualizaci, počtu objektů* a další *informace* jako jsou například *metadata*. Uvození zápatí zobrazuje obr. 4.

```
<trailer> ::= trailer                          // Definice zápatí
<<                                             // Uvození zápatí
<trailer key-value pair>+                      // Klíčové slovo
>>                                             // Ukončení zápatí
```

Obr. 4: Zápatí

1.7 Modifikace dokumentu

„Obsah *PDF dokumentu* může být modifikován bez přepsání původního souboru. Veškeré záznamy o editaci jsou zapisovány na konec souboru jako příloha původní verze. Jestliže je *PDF dokument* modifikován, jsou do nové sekce *tabulky křížových odkazů* přidány odkazy na modifikované a vytvořené objekty. Do nové sekce je také přidán odkaz na objekt číslo 0. Zároveň je na konci souboru vytvořen nový *trailer.*“ [1]

1.8 Šifrování dokumentu

Soubor PDF lze zašifrovat a chránit jej tak před neoprávněným přístupem. V dokumentu jsou šifrovány pouze znaky v objektech *string* a *stream*. Ostatní data šifrována nejsou a to důvodu zakrytí obsahové stránky dokumentu a zachování její struktury.

1.9 Metadata

Pochází z řeckého *meta* = mezi, za + latinského *dat*a = to, co je dáno. Jsou to ve své podstatě strukturovaná data o datech. Příkladem může být například katalogový lístek v knihovně, obsahující data o původu a umístění knihy v polici a sekci knihovny.

Mezi *metadata* řadíme položky typu:

- autor,
- titulek,
- obsah,
- klíčová slova,
- datum a čas vytvoření souboru,
- datum a čas poslední změny,
- verze souboru,
- program ve kterém byl soubor vytvořen.

2. Návrh a realizace projektu

Před samotným návrhem projektu (aplikace) je důležité si zvolit požadavky a cíle, kterých chceme dosáhnout.

Samotný návrh projektu by se měl zaobírat realizací projektu z hlediska:

- programovacího jazyka,
- vývojového prostředí,
- výběru a použití knihoven,
- volby komponent,
- funkcí projektu,
- vizuálního zpracování,
- funkčností.

2.1 Programovací jazyk

„*Programovací jazyk* je prostředek pro zápis *algoritmů*, jež mohou být provedeny na počítači. Zápis algoritmu ve zvoleném programovacím jazyce se nazývá *program*. Programovací jazyk je komunikačním nástrojem mezi *programátorem*, který v programovacím jazyce formuluje postup řešení daného problému, a počítačem, který program interpretuje technickými prostředky. Programovací jazyk je vlastně soubor pravidel pro zápis algoritmu, odborně řečeno se jedná o *formální jazyk*.“ [2]

Dělení programovacích jazyků podle míry abstrakce:

- vyšší programovací jazyky,
- nižší programovací jazyky.

Vyšší programovací jazyky dělíme:

- procedurální,
    - strukturované,
    - objektově orientované,
- neprocedurální,
    - funkcionální,
    - logické.

2.2 Vývojové prostředí a jeho volba

„*Vývojové prostředí* (často se používá zkratka *IDE* z anglického *Integrated development environment*) je software usnadňující práci programátorů, většinou zaměřený na jeden konkrétní *programovací jazyk*. Obsahuje *editor zdrojového kódu*, *kompilátor*, případně *interpret* a většinou také *debugger*. Některé obsahují *systém pro rychlý vývoj aplikací* (zvaný *RAD*), který slouží pro vizuální návrh grafického uživatelského rozhraní. Pokud se jedná o nástroj pro *objektově orientované programování*, může obsahovat také *object browser.*“ [3]

Vývojové prostředí si programátor volí podle *programovacího jazyka*, ve kterém aplikaci vytváří (programuje). V dnešní době je k sehnání spousta dobrých *vývojových prostředí*, jak placených tak freewarových.

Pro vytvoření tohoto projektu byly zvoleny prostředí:

- SharpDevelop 3.0,
- Microsoft Visual C# 2008 Express Edition.

2.3 Knihovny

„Knihovna - *library* - je v programování funkční logický celek, který poskytuje služby pro programy. Většinou se jedná o sbírku *procedur*, *funkcí* a *datových typů*, či při objektově orientovaném přístupu o *sadu tříd*, uložených v jednom *diskovém souboru*. *Knihovna* poskytuje *aplikační programátorské rozhraní* (zvané *API*), které umožňuje programu volat funkce poskytované touto knihovnou. Existuje mnoho knihoven pro různé účely, např. pro využívání služeb *operačního systému*, *grafické funkce*, *řízení periférií*, *vědeckotechnické výpočty* atp.“ [4]

Pro projekt bylo potřeba využít *knihoven*, které by umožnily načtení textu, počtu stran dokumentu a *metadat* souboru. K tomuto účelu byla nejprve vybrána knihovna *iTextSharp*, která všechny uvedené parametry splňuje. Od této volby bylo později částečně upuštěno z důvodu problémového načítání textu. Tato knihovna načítala text jako pole znaků, což vedlo k zjištění nekompatibility s *ASCII tabulkou Win1250 (CP-1250)*. U této knihovny se totiž projevily chyby při načítání některých znaků s diakritikou (tj. háčky nad písmeny a kroužek nad písmenem u). Tato knihovna byla úspěšně použita pro načtení *metadat* ze souboru.

Pro načítání textu ze souboru byla místo již zmíněné knihovny použita knihovna *PDFBox-0.7.3*. Tato knihovna nečte text jako pole znaků, ale jako řetězec, čímž bylo zajištěno bezproblémové čtení.

Pro načtení textu ze souboru s příponou pdf byl použit sled příkazů (Obr. 5).

```
using org.pdfbox.pdmodel;                                    // Propojení knihovny
using org.pdfbox.util;                                       // Propojení knihovny

PDDocument doc = PDDocument.load(soubor);                    // propojení souboru
PDFTextStripper pdfStripper = new PDFTextStripper();         // funkce pro převod
string PDF_string = pdfStripper.getText(doc);                // načtení textu do promněné
```

Obr. 5: Načtení textu pomocí knihovny PDFBox-0.7.3.

Při použití knihovny *PDFBox-0.7.3.* může nastat problém při načítání textu formátovaným do bloku, neboť při tomto formátování se v textu objevuje několik mezer přímo za sebou. Tohoto paradoxu se musíme zbavit kvůli zjištění přesného počtu znaků v textu a hlavně kvůli případnému vyhledávání. Tento problém byl vyřešen následným sledem příkazů (Obr. 6).

```
for (int pomx = 0; pomx <= PDF_string.Length - 2; pomx += 1)        // procházení znaků
                                                                       načteného textu
    if (PDF_string[pomx].ToString() != " ")                         // kontrola znaku
      PDF_string_new = PDF_string_new + PDF_string[pomx];
    else
    {
    PDF_string_new = PDF_string_new + " ";
    while (PDF_string[pomx + 1].ToString() == " ") pomx += 1;       // cyklus ke kontrola
                                                                       dalšího znaku
    }
```

Obr. 6: Korekce mezer

2.4 Komponenty

Komponenta je předpřipravený funkční celek, s přesně definovaným rozhraním, která může být nezávisle použita na prostředí, ve kterém nebo pro které byla naprogramována.

V tomto projektu, kromě standardně používaných komponent, byla použita nestandardní komponenta *GridViewBeta2*. Tato komponenta byla použita pro zobrazení výsledků kontroly, neboť mezi standardními *komponentami* nebyla komponenta podobající se tabulce, do které by mohly být zobrazeny výsledky, nalezena.

3. Program pro ověření a kontrolu dokumentů s příponou pdf

Program můžeme rozdělit do čtyř částí:

- výběr dokumentů určených pro kontrolu,
- zadání a kontrola splněných podmínek,
- přehled výsledků kontroly,
- výstup.

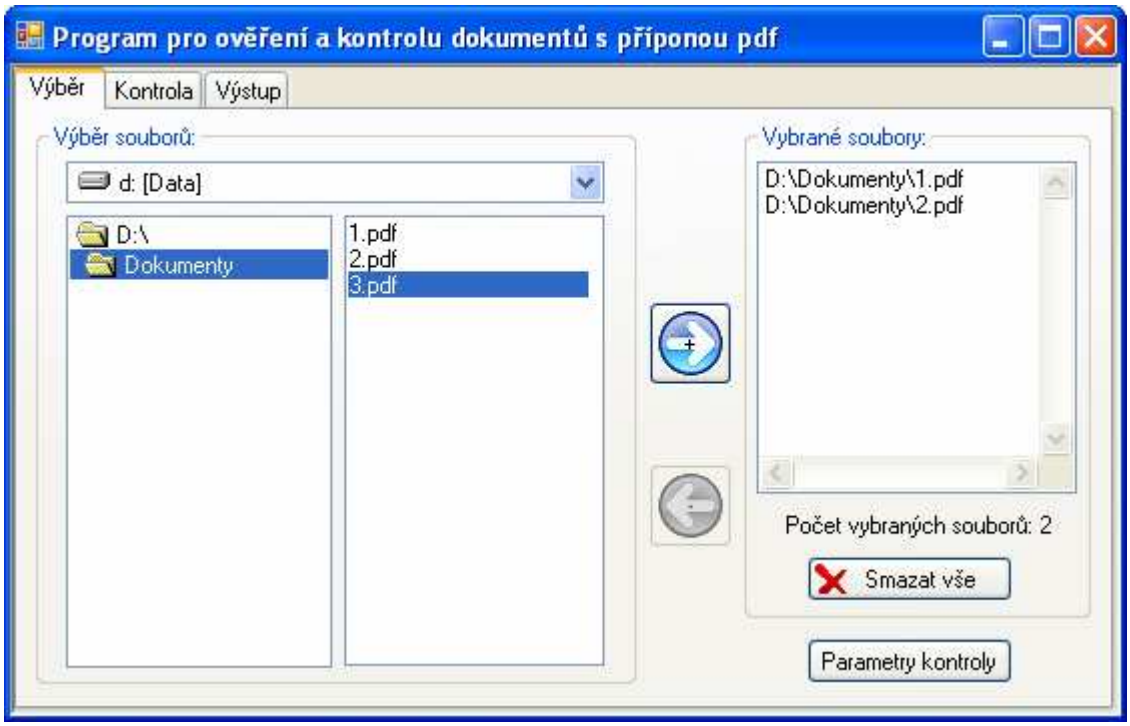
3.1 Výběr souboru

Při spuštění programu se uživateli zobrazí okno pro výběr dokumentu k ověření a kontrole PDF souboru (Obr. 7). Toto okno je rozděleno do dvou boxů:

- *Výběr souboru:*,
- *Vybrané soubory*:.

V boxu *Výběr souboru:* si uživatel vybere soubory určené pro kontrolu a ověření. Dvojklikem nebo jedním klikem tlačítka myši na název souboru a následným potvrzením výběru stiskem tlačítka pro přidání, se přidá soubor. Kompletní cesta s názvem vybraného souboru se zobrazí v boxu *Vybrané soubory*:. Tímto uživatel získá přehled o již vybraných souborech a jejich umístění.

V boxu *Vybrané soubory*: se soubory zobrazují s kompletní cestou jejich umístění z důvodu zabránění vzniku chyb ve výběru, způsobených stejným pojmenováním souborů s různým umístěním a tím pádem potenciálně různých. Pokud uživatel chce z boxu *Vybrané soubory:* některé soubory odebrat, postupuje principiálně stejně, jako při jejich výběru. Pro přechod k volbě *kontrolních bodů* a *podmínek* uživatel klikne na tlačítko *Parametry kontroly*, nebo se pomocí myši přepne na záložku *Kontrola*.

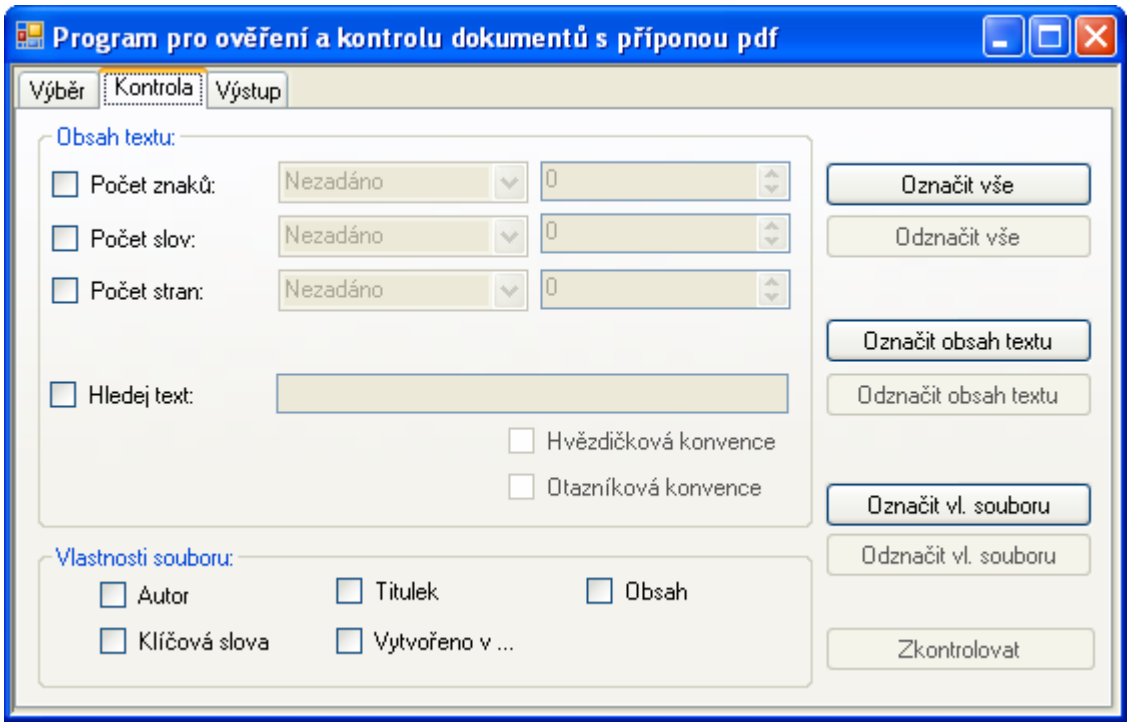


Obr. 7: Okno pro výběr souboru

3.2 Zadání a kontrola splnění zadaných podmínek

Kontrolní body a *podmínky* uživatel zadává na záložce *Kontrola* (Obr. 8). Na této záložce má uživatel k dispozici dvě skupiny kontroly, které se zajímají o kontrolovaný soubor z různých hledisek. Skupiny:

- *Obsah textu:*,
- *Vlastnosti souboru:*.



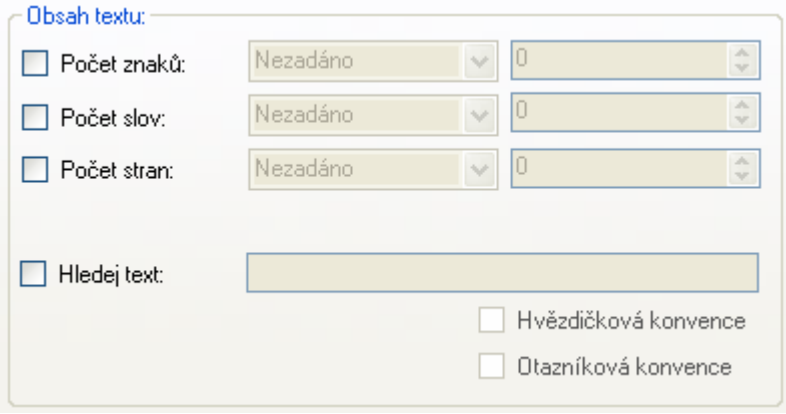
Obr. 8: Záložka Kontrola

3.3 Obsah textu:

Tato skupina se o kontrolu souboru zajímá z hlediska obsahu. Uživatel zde má možnost zvolit z několika bodů (Obr. 9), u kterých může přesněji určovat, co se má v souboru přesně ověřit.

Body kontroly:

- *Počet znaků:*,
- *Počet slov:*,
- *Počet stran:*,
- *Hledej text:*.

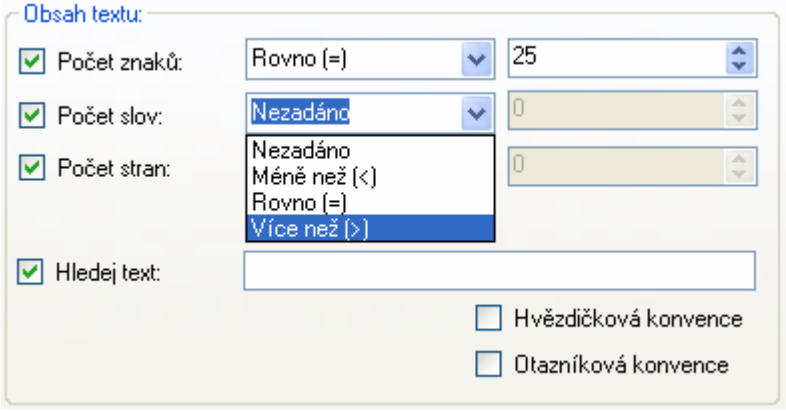


Obr. 9: Podmínky kontroly

Po zvolení bodu kontroly dostane uživatel možnost nastavit body ověření pro upřesnění hledaných parametrů (Obr. 10).

Body ověření:

- *Nezadáno,*
- *Méně než (<),*
- *Rovno (=),*
- *Více než (>).*



Obr. 10: Podmínky ověření

Při zadání jiného parametru než *Nezadáno*, bude moci uživatel zadat přesný počet hledaných prvků, podle kterých se řídí správnost odpovědi na výstupu.

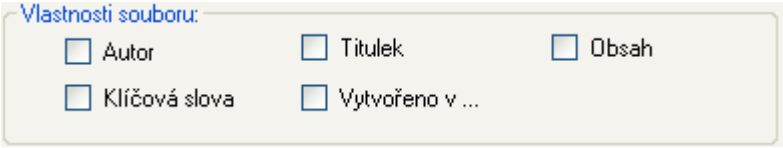
U bodu kontroly *Hledej text …* může uživatel používat *hvězdičkovou* a *otazníkovou konvenci*, což mu umožňuje definovat dynamické podmínky hledání.

3.4 Vlastnosti souboru:

Tato kategorie, jak název napovídá, se nezajímá o text a obsah, ale zajímá se o soubor jako takový. Výstupem této kategorie jsou informace o *autoru souboru* (Obr. 11).

Jsou to:

- *Autor*,
- *Titulek*,
- *Obsah*,
- *Klíčová slova*,
- *Vytvořeno v ….*



Obr. 11: Vlastnosti souborů

Všechny tyto informace jsou u souboru nepovinné, to znamená, že nemusí být vyplněny. Jsou získány z tak zvaných *Metadat*. Pro spuštění kontroly uživatel pokračuje volbou tlačítka *Zkontrolovat*.

3.5 Zobrazení výsledků kontroly

Výsledky kontroly jsou zpracovány *do tabulky výsledků* (Tab. 2) s možností uložení dat do *souboru XML* pro pozdější zobrazení, popřípadě další zpracování. V tabulce jsou body ověření *Méně než (<), Rovno (=), Více než (>) a Hledat text* reprezentovány výrazy *ANO / NE* podle splnění zadaného bodu. Dále se u těchto bodů zobrazuje parametr *Počet* jako vizuální kontrola pro ověření správnosti výsledku. Bod ověření *Nezadáno* je reprezentován číslem, jako přesný výsledek na B*od kontroly* a u položky *Hledej text:* hledaným textem (řetězcem znaků).

Tab. 2: Tabulka zobrazení výsledků

Název souboru s cestou	Autor:	Martin Sohr				
	Titulek:	Bakal. práce				
	Obsah:	Bakal. práce				
	Klíčová slova:	PDF, XML….				
	Vytvořeno v:	xxxx				
	Body ověření	Nezadáno	<	=	>	Počet
	Počet znaků:	1000	ANO			10000
	Počet slov:	200		NE		222
	Počet stran:	5			NE	8
	Hledej text:	TEXT		ANO		

3.6 Report

Výsledný report je pro budoucí zpracování uložen do stromové struktury souboru *XML*.

3.6.1 XML (Extensible Markup Language)

Jazyk XML představuje způsob vložení strukturovaných dat (například informací v listu) do textového souboru, který se řídí určitými standardy a který lze číst pomocí velkého množství aplikací. V kódu *XML* je možno vytvářet vlastní značky (*tzv. tagy*), což umožní popisovat a interpretovat data mezi aplikacemi. *XML soubor* není definován binárně, takže je dobře čitelný i pro uživatele. Výstižné a inteligentní pojmenování tagů, umožní uživateli lepší pochopení vytvoření struktury.

3.6.2 Ukázky stromové struktury

Formát uložených dat výsledku kontroly ve zkrácené stromové struktuře viz Obr. 12 a rozložené struktury jednoho uloženého *bodu kontroly* viz Obr. 13.

```
−<Seznam_souboru>
        −<soubor>
            D:\škola\szz\bakalařka\Sohr - Bakalářská práce 1.9.pdf
            −<meta_data>
                +<Autor></Autor>
                +<Titulek></Titulek>
                +<Obsah></Obsah>
                +<klicová_slova></klicová_slova>
                +<vytvořeno_v_programu></vytvořeno_v_programu>
             </meta_data>
             −<vlastnosti_souboru>
                +<Počet_znaku></Počet_znaku>
                +<Počet_slov></Počet_slov>
                +<Pocet_stran></Pocet_stran>
              +<Hledej_text></Hledej_text>
             </vlastnosti_souboru>
        </soubor>
</Seznam_souboru>
```

Obr. 12: Ukázka zkrácené stromové struktury XML

```
−<Autor>                                    // Tag
  Autor:                                    // Popisek po tagu
  <Vysledek>                                // Tag
    Martin                                  // Výsledek
  </Vysledek>                               // Ukončení tagu
</Autor>                                    // Ukončení tagu
```

Obr. 13: Ukázka jednoho kontrolního bodu uloženého v XML

Pokud uživatel nekontroluje soubor vzhledem k nějakému *kontrolnímu bodu*, tak se tento bod do stromové struktury neuloží.

Příklad vytvoření stromové struktury XML z Obr. 12 v programovacím jazyce C# (Obr. 14).

```
using System.Xml;                                   // Nastavení knihovny
using System.Xml.Xsl;                               // Nastavení knihovny

XmlTextWriter tw = new XmlTextWriter (Název soubor,
System.Text.Encoding.BigEndianUnicode);             // Vytvoření souboru XML
tw.WriteStartDocument();                            // Otevření souboru
        tw.WriteStartElement(Název tagu);           // Začátek nové větve (Autor)
                tw.WriteString(Popisek);            // Popisek větve (Autor:)
                tw.WriteStartElement(Název tagu 2); // Začátek nové větve (Vysledek)
                        tw.WriteString(Popisek);    // Popisek větve (Hodnota)
                tw.WriteEndElement();               //Konec větve (Vysledek)
        tw.WriteEndElement();                       //Konec větve (Autor)
tw.Close();                                         //Uzavření souboru
```

Obr. 14: Vytvoření stromové struktury v C#

3.6.3 Předloha stylů – XSL jazyk

Předloha se styly je správně vytvořený dokument *XML* využívající k převedení informací ze soubor *XML* na soubor zvolený pomocí zvláštní sady pokynů. Použijete-li předlohu se styly u souboru *XML*, bude tato předloha řídit formátování nebo způsob zobrazení tohoto souboru. Standard předloh se styly jazyka XML je označován jako *jazyk XSL (Extensible Style Language).*

XLS - *eXtensible Stylesheet Language* – kaskádový styl (CSS), který umožňuje definovat vzhled jednotlivých elementu souboru s příponou XML jakou jsou:

- zarovnání,
- velikost písma,
- styl písma,
- barvy,
- formátování.

4. Závěr

Cílem práce bylo vytvořit *Program pro ověření a kontrolu dokumentů s příponou pdf* v programovacím *jazyce C#.* Pravidelnými konzultacemi s vedoucím projektu byl vytvořený program směřován podle představ a připomínek vedoucího projektu. Pro programování byl použit *freewarový* program *SharpDevelop 3.0* a *Microsoft Visual C# 2008 Express Edition.*

Vytvořený program dává uživateli možnost zkontrolovat PDF dokument vzhledem k obsahu a informacím o souboru. Uživateli jsou k dispozici přeprogramované funkce, jako například určení počtu znaků, slov, stran dokumentu nebo vyhledávání textu (řetězce znaků). Dále uživatel může získat informace o autoru souboru jako jsou jméno a příjmení, titulek, název a verzi programu, ve kterým byl dokument vytvořen, klíčová slova a další parametry.

Při načtení poškozeného souboru s příponou pdf se zobrazí hláška o neošetřené výjimce. Tato výjimka byla ošetřena příkazy *TRY -> CATCH*, ale z neznámého důvodu není vyvolána při objevení chyby.

5. Literatura

[1] KRAUS, Vladimír. Vladimír Kraus: PDF : 4. Struktura dokumentu [online]. 08.06.1999 11:18. 08.06.1999 11:18 , 08.06.1999 11:18 [cit. 2008-12-15]. Dostupný z WWW: <http://www-kiv.zcu.cz/~herout/html_sbo/pdf/4.htm>.

[2] Programovací jazyk [online]. 19. 5. 2009 v 20:03. Wikipedie, 19. 5. 2009 v 20:03 , 19. 5. 2009 v 20:03 [cit. 2009-05-24]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Programovací_jazyk>.

[3] Vývojové prostředí [online]. Stránka byla naposledy editována 18. 5. 2009 v 09:34. Wikipedie, 18. 5. 2009 v 09:34 , 18. 5. 2009 v 09:34. [cit. 2009-05-24]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Vývojové_prostředí>.

[4] Knihovna programování [online]. Stránka byla naposledy editována 11. 5. 2009 v 10:46. Wikipedie, 11. 5. 2009 v 10:46 , 11. 5. 2009 v 10:46. [cit. 2009-05-24]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Knihovna_(programování)>.

6. Seznam zkratek

C#	Programovací jazyk
OS	Operační systém
CSS	Kaskádový styl
ISO	Norma
PDF	Portable Document Format (Přenositelný formát souboru)
ASCII	American Standard Code for Information Interchange
UNIX, Windows, Mac	Operační systémy
ZIP	Kompresní formát
JPEG2000	Formát ztrátové komprese
Open Type	Popis vektorových písem – fontů
3D	Trojrozměrný prostor
IDE	Vývojové prostředí
RAD	Systém pro rychlý vývoj aplikací
API	Aplikační programátorské rozhraní
WIN1250,CP1250	Kódování ASCII tabulky
XML	Rozšiřitelný značkovací jazyk
XSL	Rozšiřitelný stylový jazyk